TRATAMIENTO DEL SEÑAL POR COMPUTADOR

DETECCIÓN DE FALLOS MECÁNICOS MEDIANTE ANÁLISIS ESPECTRAL

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Sergio López Cantero
i dirigit per
Pedro Balaguer Herrero
Bellaterra 15 de juny de 2007

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Pedro Balaguer Herrero

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Sergio López Cantero

I per tal que consti firma la present.

Signat: .....

Bellaterra, 15 de juny de 2007

-Índice-

1. Introducción

El uso del análisis de las vibraciones mecánicas como método de mantenimiento predictivo es una técnica que lleva siendo empleada con éxito desde hace muchos años en la industria.
En un principio solo se utilizaba de forma limitada -años ochenta-, puesto que las máquinas de la época no permitían mayor procesado de datos, dichas máquinas eran ordenadores PC 8086 u 80286.
Pero desde entonces los equipos informáticos han mejorado mucho, pudiéndose realizar hoy día estudios del espectro y análisis predictivos mucho más complejos. Y no sólo en la capacidad de cálculo, sino también en la facilidad de uso de los mismos.
Sin embargo, la metodología sigue siendo muy similar: las máquinas que realizan tareas de bajo riesgo reciben un control periódico, con análisis en intervalos de tiempo decrecientes a medida que su tarea es mas crítica –ya sea por peligro de los trabajadores o por la función que desempeña la máquina en la cadena de fabricación-. Llegando a controlarse de forma continua aquellas máquinas con unas tareas más críticas.
La última revolución son las Tecnologías de la Información, que permiten un control remoto del estado de las máquinas, este control remoto se puede realizar ya desde otras partes de la misma fábrica a través de cableado interno o mediante conexión a Internet desde oficinas sitiadas a cualquier distancia.

En este proyecto se pretende ir un paso mas allá; en lugar de estudiar las vibraciones, se estudiará el sonido que emiten las máquinas para poder detectar fallos en el funcionamiento de las mismas y se pretende crear una aplicación que funcione en un PC estándar.

1.1 - Objetivo

El objetivo de este proyecto es crear un método y una aplicación que ayudaran a resolver el problema de la detección de fallos mecánicos. La solución al problema fue evolucionando en la realización de este proyecto, llegando a ella mediante hipótesis y sus comprobaciones tanto teóricas como prácticas.
Dado que el problema no es de solución única, pero se ha llegado a una solución eficiente y simple.

1.2 - Motivación

La detección de fallos es el estudio del comportamiento para detectar mediante sensores en las máquinas –en nuestro caso eléctricas- su correcto funcionamiento.

La detección de fallos es algo muy importante de cara al trabajo industrial, sobre todo en maquinaria pesada, ya que puede ayudar a prevenir accidentes y hasta la pérdida de vidas.
También tiene un peso importante en el mantenimiento de las máquinas al poder tener de manera inmediata un indicador del estado de las mismas.
Gracias a esto último, un gran número de ellas pueden ser mantenidas por pocas personas.
Este proyecto pretende estudiar otros métodos de la detección de fallos, a través de un ejemplo teórico-practico. El programa elaborado deberá ser capaz de llevar a la práctica las técnicas de detección que en este trabajo se han creado.

1.3 – Características

Las siguientes características forman parte del proyecto realizado, se hablará de todas ellas a lo largo de esta memoria.

-Adquisición de señal vía tarjeta de sonido.
Uso del conversor Analógico/Digital de la tarjeta de sonido para adquirir la señal sonora desde el micrófono.

-Cálculo de la FFT.
Cálculo del espectro de la señal sonora adquirida mediante la implementación de la transformada rápida de Fourier.

-Cuantificación de la señal en bandas de energía.
Partición del espectro calculado en 10 bandas de energía para su análisis estadístico.

-Análisis estadístico de los datos.
Análisis estadístico de la distribución normal de los datos, calculando su media y desviación estándar para su uso posterior.

-Evaluación del estado de la máquina a través de los datos estadísticos
Utilizando los datos calculados previamente y un test de hipótesis a cada banda de frecuencias, se decide el estado de la máquina según una nueva señal sonora adquirida.

-Experimentación con motor DC.
Realización de experimentos con un motor real de corriente continua para comprobar el correcto funcionamiento de las herramientas desarrolladas.

2. Fundamentos teóricos

2.1 - Detección de fallos

La detección de fallos por computador es un campo que busca la manera de automatizar el proceso de detectar fallos en las máquinas de manera que no haga falta un supervisor in-situ, solo un operario que puede controlar varias máquinas a la vez mediante un PC a distancia.

2.2 – Análisis de Fourier

En este tema se tratarán diferentes partes teóricas relacionadas con el tratamiento de las señales discretas, entre ellas la transformada de Fourier.

2.2.1 - Introducción a las señales discretas

Las señales discretas, a diferencia de las continuas, no están definidas en todo el tiempo ni pueden tomar cualquier valor dado que están cuantizadas. La cuantización es el proceso de seleccionar los valores que pueden tomar las muestras, estos dependen de la resolución en bits que se tome para representar sus magnitudes, a mayor número de bits mayor precisión de la magnitud.
Las muestras son las representaciones numéricas de la señal continua en un instante de tiempo determinado.
La cantidad de muestras en el tiempo, sin embargo, viene definida por la frecuencia de muestreo (Fm) o el tiempo de muestreo (Tm) - 1/Tm = Fm – definiendo cada cuanto tiempo se adquirirá una muestra de la señal continua.
En la figura 2.1 podemos ver que la señal no esta definida en todos los puntos, la separación entre muestras viene definida por la frecuencia de muestreo, la cuantización limitaría la mínima distancia vertical entre 2 posibles valores de una muestra.

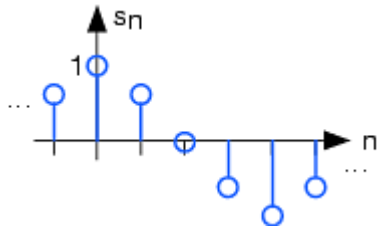

Figura 2.1 Señal discreta de un Coseno

2.2.2 - Transformada de Fourier Discreta

La transformada de Fourier es una formula que transforma una señal temporal en una frecuencial, representando la señal en forma de un sumatorio de otras funciones, en este caso exponenciales.
La Transformada de Fourier Discreta (DFT) calcula el espectro de una señal discreta en N componentes frecuenciales siendo N el número de muestras de la señal discreta.
El espectro de una señal representa a la misma como una suma de frecuencias indicando en que frecuencias la señal tiene más energía, es decir, donde sus componentes frecuenciales son mayores.
En nuestro caso utilizamos la transformada discreta ya que la señal sonora adquirida también es discreta después de ser muestreada.
En la *Ecuación 2.2* podemos ver la fórmula de la transformada de Fourier discreta, donde f_j representa el coeficiente de la señal en la posición j del espectro, x_k representa la señal discreta en el instante k, n es el número de muestras de la señal discreta.

$$f_j = \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \qquad j = 0, \ldots, n-1.$$

Ecuación 2.2 - Transformada de Fourier discreta

2.2.3 - FFT (Fast Fourier Transform)

La FFT es un algoritmo que implementa la DFT que permite reducir el tiempo de cálculo de $O(N^2)$ a $O(N*\log_2 N)$ disminuyendo considerablemente el tiempo de cálculo, especialmente con secuencias largas.
Para aclarar estos términos, en el caso concreto de este proyecto se probaron ambas implementaciones, con unas señales de 4000 muestras, la implementación de la DFT según la Ecuación 2.2, resultaba en unos tiempos de computación que rondaban los 30 segundos, mientras que la FFT tiene un tiempo de cómputo inferior a un segundo, y la diferencia se acentúa aún mas si se aumentan las muestras, ya que el incremento de la DFT es cuadrático, pudiendo llegar a más de una hora en el caso de señales de 20.000 o 40.000 muestras.

Como parte negativa, las señales deben tener un número de muestras potencia de dos pero esto se puede corregir rellenando la señal con ceros.

2.2.4 - Teorema de Nyquist

El teorema de Nyquist afirma que cuando se muestrea una señal, la frecuencia de muestreo debe ser mayor que dos veces el ancho de banda de la señal de entrada, para poder reconstruir la señal original de forma exacta a partir de sus muestras.
En nuestro caso el teorema de Nyquist nos dice que la frecuencia de muestreo seleccionada para la máquina que queremos analizar, debe ser el doble que el ancho de banda de la misma, este valor dependerá de la máquina en cuestión y se deberá realizar el cálculo previamente -o bien empíricamente o bien por las especificaciones de la máquina- para seleccionar una frecuencia de muestreo que muestree nuestra señal sonora adecuadamente.

2.2.4 - Relación de Parseval discreta

En la *Ecuación 2.3* se puede ver dicha relación, siendo x[n] el espectro de la señal en el punto n y X[k] la señal en el punto k.

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

Ecuación 2.3 – Relación de Parseval Discreta

La energía total de la señal *x*[n] es igual a la energía total de su transformada de Fourier *X[k]* a lo largo de todas sus componentes frecuenciales.
La relación de Parseval permite asegurar que el uso de la energía espectral es correcta dada su relación con la señal original.

2.3 - Estadística

2.3.1 - Control estadístico de calidad

El control estadístico de calidad es un área donde se utilizan análisis estadísticos para determinar la calidad de procesos o máquinas, en el caso de este proyecto se analiza a que clase de señal de las aprendidas pertenece la nueva señal adquirida utilizando los las medias y desviaciones estándar de cada banda de frecuencia calculados previamente mediante el aprendizaje del sistema frente a la repetición de un mismo tipo de señal.

2.3.2 - Distribución normal

La distribución normal, también llamada distribución de Gauss o distribución *gaussiana*, es la distribución de probabilidad que con más frecuencia aparece en estadística y teoría de probabilidades. Esto se debe a dos razones fundamentalmente:
- Su función de densidad es simétrica y con forma de campana, lo que favorece su aplicación como modelo a gran número de variables estadísticas.
-Es límite de otras distribuciones y aparece relacionada con multitud de resultados ligados a la teoría de las probabilidades gracias a sus propiedades matemáticas.

En la *Figura 2.4* podemos ver representada la gráfica de una distribución normal, el punto central representa la media de los datos, los intervalos de diversos multiplicadores de sigma o σ, nos indican el porcentaje de muestras que pertenecen a ese intervalo [media-σ, media+ σ]

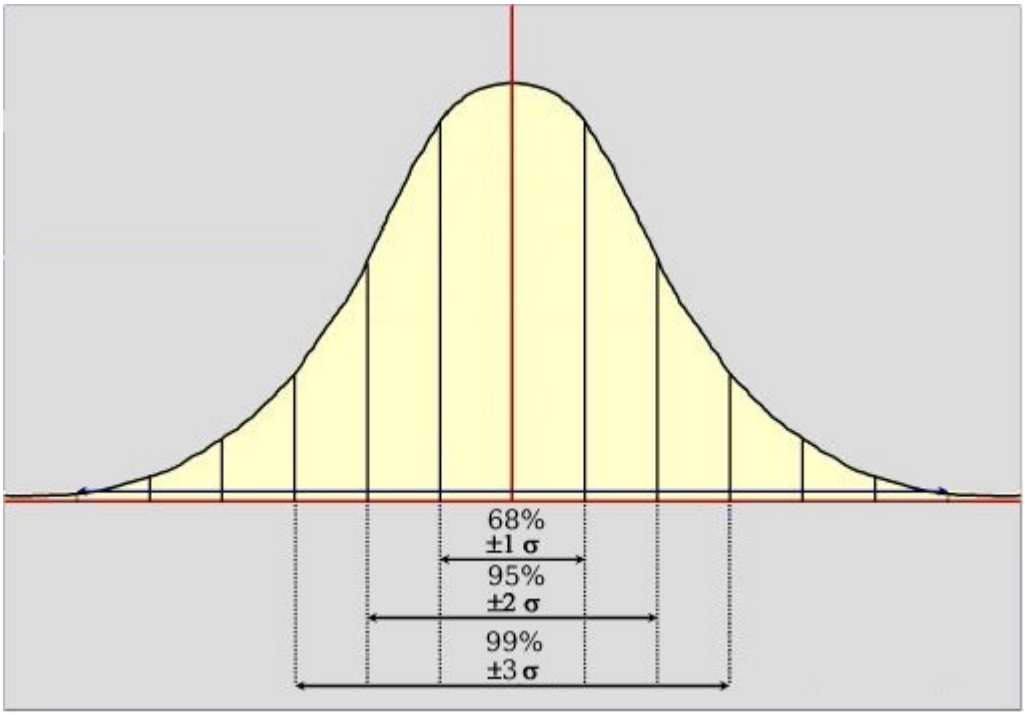

Figura 2.4 – Distribución normal estándar

Utilizando el modelo de probabilidad de la distribución normal, la media y desviación estándar tendremos un modelo estadístico para discernir si la señal de entrada es correcta o no mediante el test de hipótesis.

2.3.3 - Test de Hipótesis

Un contraste o test de hipótesis es una técnica de Inferencia Estadística que permite comprobar si la información que proporciona una muestra observada concuerda -o no- con la hipótesis estadística formulada sobre el modelo de probabilidad en estudio.
Una hipótesis estadística es cualquier conjetura sobre una o varias características de interés de un modelo de probabilidad.
El test de hipótesis tiene como objetivo rechazar o aceptar hipótesis.

En la *Figura 2.5* se puede observar un test de hipótesis donde se quiere aclarar si la muestra cae dentro de nuestro intervalo de confianza -[media-3*σ, media+ 3*σ] -, en este caso se puede ver como cae en este intervalo, al cual pertenecen el 99% de las muestras de la distribución.

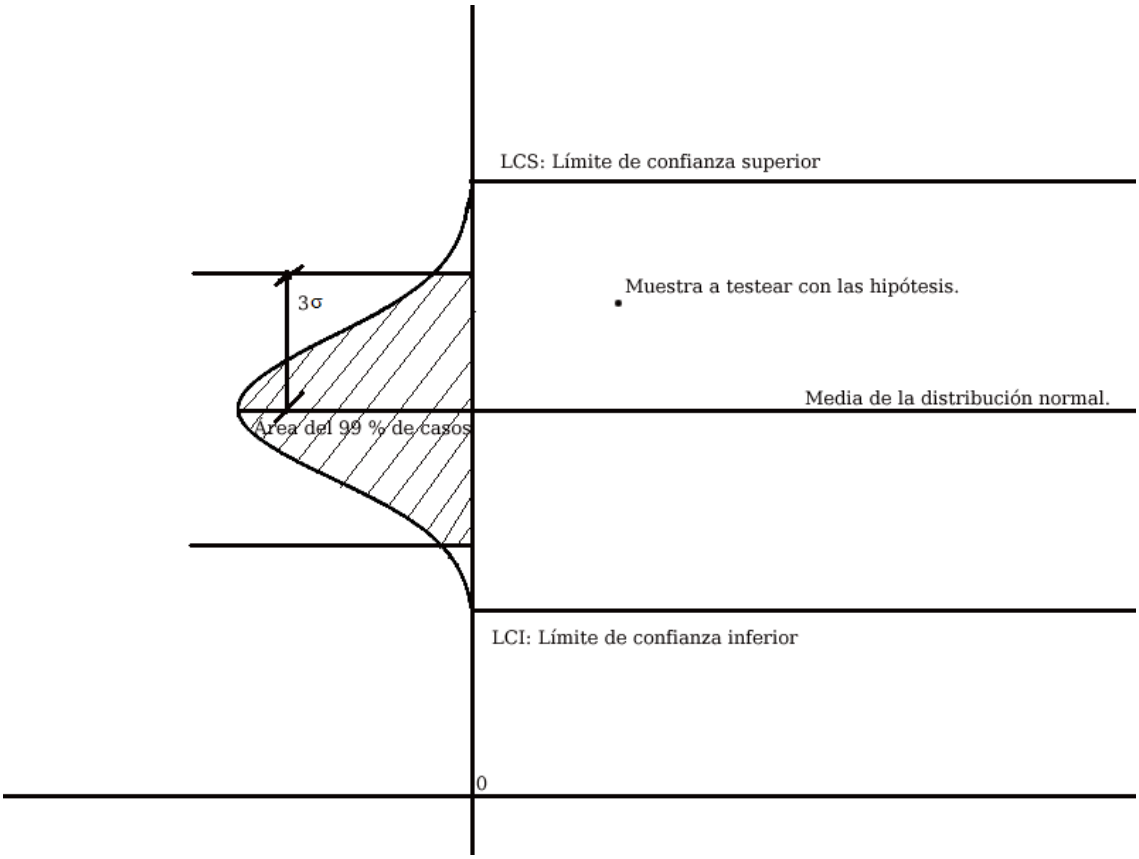

Figura 2.5 – Test de hipótesis sobre la variable del eje de ordenadas

Hay 4 posibilidades en un test de hipótesis:

	H0 es cierta	H1 es cierta
Aceptada H0	No hay error.	Error.
Aceptada H1	Error.	No hay error.

En nuestro caso, suponiendo dos hipótesis, H0 y H1, siendo H0 el echo que la variable perteneciente al porcentaje de una banda de frecuencias pertenezca a la distribución normal y H1 que no pertenezca, el test de hipótesis establece el porcentaje de que ocurra una cosa u otra con un porcentaje de fiabilidad, tal como se puede ver en la *Figura 2.4*: [media-3*σ, media+3*σ] Este intervalo define el 99% de las muestras de H0, de modo que si nuestra variable cae en tal intervalo, hay un 99% de posibilidad de que la hipótesis sea cierta, es decir, de que nuestra variable aleatoria pertenezca a dicha distribución.

2.3.4 – Variables Independientes e Idénticamente Distribuidas (IID)

Asimismo asumimos que las variables son IID o Independientes Idénticamente Distribuidas, o lo que es lo mismo, cada una tiene la misma distribución de probabilidad y son mutuamente independientes.

3. Solución presentada

Primero se explicará la solución adoptada para más tarde ir presentando cada una de sus partes tanto teórica como práctica.
La solución al problema tiene dos partes: por un lado se sitúa la adquisición y tratamiento de la señal, en el cual se adquiere, vía micrófono, una señal y se le aplica la transformada de Fourier para obtener su espectro.
A la hora de realizar el análisis estadístico de la señal que se quiere aprender, el sistema toma varias muestras de un mismo tipo de señal acústica –un mismo estado de la máquina- de forma continua, calcula el espectro de cada una para posteriormente calcular la media y desviación estándar de esta clase que se quiere aprender.
A la hora de evaluar si una nueva señal acústica pertenece a nuestras clases aprendidas con la aplicación, se utilizará un algoritmo propio que hará uso de los test de hipótesis para decidir si la nueva señal forma parte de las clases aprendidas determinando un estado de la máquina o por otro lado la señal analizada no tiene relación con estas clases y es una señal externa.

3.1 - Bases teóricas utilizadas

La solución al problema propuesta hace uso de tres grandes pilares:

3.1.1 – Análisis de Fourier

Para el paso de una señal temporal al dominio frecuencial y el posterior análisis de la señal, es imprescindible el uso de la transformada de Fourier, en el caso concreto de esta aplicación se utilizó la FFT o Fast Fourier Transform, ya que tiene una complejidad de orden $O(N*\log_2 N)$ mientras que la transformada de Fourier normal tiene un orden $O(N^2)$.

3.1.2 – Energía

La concentración energética de la señal dividida en 10 bandas de energía será el valor que se utilizará para los cálculos estadísticos, el teorema de Parseval nos dice que la concentración de energía del espectro es la misma que la de la señal, por lo tanto para una banda de frecuencias tenemos una concentración energética correspondiente a la señal que representan dichas frecuencias. De esta manera analizamos la señal de entrada como varias señales independientes a cada una de las cuales realizamos un estudio estadístico y que juntas forman la señal de entrada.

3.1.3 – Estadística

La asunción de una distribución normal –vistos los histogramas de las señales- nos permite realizar un estudio estadístico sobre el espectro de las señales.
Para ello se utilizan los dos parámetros que definen la normal, la media aritmética y la desviación estándar, con estos dos parámetros se establece un intervalo donde se prueba si la banda de frecuencia pertenece a la señal o no.

3.2 - Aplicación práctica de las bases teóricas utilizadas

Estas bases teóricas mostradas serán utilizadas para alcanzar los diversos objetivos del proyecto de ahora en adelante.

3.2.1 - Cálculo del Espectro de una señal

Mediante la FFT se transformaran las señales de tiempo discreto al espacio frecuencial, donde podrán ser analizadas estadísticamente.
El algoritmo de la FFT se basa en partir la señal en subseñales de longitud N/2 sucesivamente hasta llegar a una longitud pequeña para calcular su espectro y después reconstruir el espectro original utilizando los mas pequeños.
En el apéndice 7.2.1 se puede ver la implementación en C/C++ de la FFT utilizada en el proyecto.

3.2.2 - Cálculo de distribuciones normales del conjunto de señales a aprender

El análisis de varias señales muestreadas adquiridas permitirá realizar un estudio estadístico de las mismas a través del cálculo de su media y desviación estándar, estos datos se utilizaran en el test de hipótesis para identificar si los nuevos espectros pertenecen a alguna de las clases aprendidas por el programa.

3.2.3 - Capacidad de poder evaluar nuevas señales comparándolas con las aprendidas

La aplicación evaluará las entradas de señales nuevas adquiridas mediante la tarjeta de sonido para decidir a que tipo de señal aprendida pertenece utilizando los datos adquiridos mediante el análisis estadístico.

3.2.4 – Adquisición de la señal mediante el micrófono

Para la adquisición de las muestras de sonido se han utilizado las librerías de Windows, escritas en C++: Waveform Audio

Esta tabla es una referencia sobre las funciones utilizadas en el programa, organizadas en el orden en el que se llaman para ser utilizadas. De modo que establece un flujo cronológico de las acciones a realizar para capturar audio.

MMRESULT: resultado de la operación devuelto por las funciones de la biblioteca, según su valor sabremos si la operación ha sido un éxito o si ha fallado, cual es el motivo.
HWAVEIN: estructura que contiene las especificaciones de la entrada de audio.
WAVEHDR: estructura para el buffer de audio.
WAVE_MAPPER: constante que define la interfaz de audio a utilizar.

MMRESULT openAudioIn(HWAVEIN *pAudioInH, int nSamplesPerSec, int nBitsPerSample);	Inicializa el sistema de adquisición de sonido con el número de bits por muestra y muestras por segundo especificado. Esta función es creada por mí para un acceso más fácil a la siguiente función.
MMRESULT waveInOpen(pAudioInH, WAVE_MAPPER, &fmt, 0, NULL, CALLBACK_NULL);	Función de apertura del interfaz de adquisición de datos, el último parámetro puede especificar una función a la que se llama si el buffer de recepción se llena por completo.
MMRESULT waveInPrepareHeader(hWaveIn, &wavedata,sizeof(WAVEHDR));	Prepara la estructura para recibir los datos de la tarjeta de sonido
MMRESULT waveInStart(hWaveIn);	Comienza la adquisición de datos.
MMRESULT waveInAddBuffer(hWaveIn, &wavedata,sizeof(WAVEHDR));	Añade un buffer a la entrada de datos para poder guardar los datos que se adquieren.
waveInStop(hWaveIn);	Para la adquisición de datos.
waveInReset(hWaveIn);	Reinicia el sistema de audio.
waveInClose(hWaveIn);	Cierra el sistema de audio.

3.3 - Diseño de algoritmos

En este capítulo se cubrirán los dos algoritmos básicos para la realización de este proyecto, un algoritmo para poder aprender que señales queremos identificar con nuestra aplicación y otro para una vez aprendidas, poder decidir si nuevas señales acústicas que entremos en el sistema pertenecen a esas clases aprendidas previamente o por el contrario son señales ajenas al aprendizaje.

3.3.1 - Método de aprendizaje

Para poder clasificar las nuevas señales que se adquieren y así decidir el estado de la máquina de la que se esta adquiriendo el sonido, es necesario que el sistema "aprenda" los diferentes tipos de señal que queremos diferenciar. El método utilizado es el siguiente:
Se captura varias veces el mismo tipo de señal que se quiere aprender de forma continua y se calcula el espectro mediante la FFT de cada una de estas señales, estos espectros se dividen en 10 bandas de frecuencia según la energía de la señal de cada banda de frecuencias.

En la *Figura 3.1* se puede ver la concentración de energía de cada banda del espectro, estas representan los |Ck| para cada rango de frecuencias.

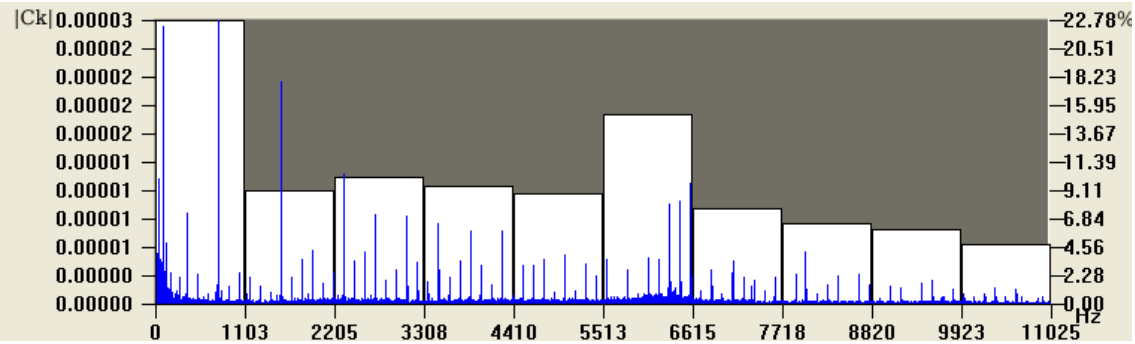

Figura 3.1 – Representación en bandas de energía del espectro de una señal

En cada banda se calcula la media de la energía y su desviación estándar. Utilizamos la desviación estándar porque asumimos una distribución normal de los datos, aquí podemos ver las gráficas que representan el histograma de las bandas a 1.2, 2.4 y 3.6V en la *Figura 3.2*

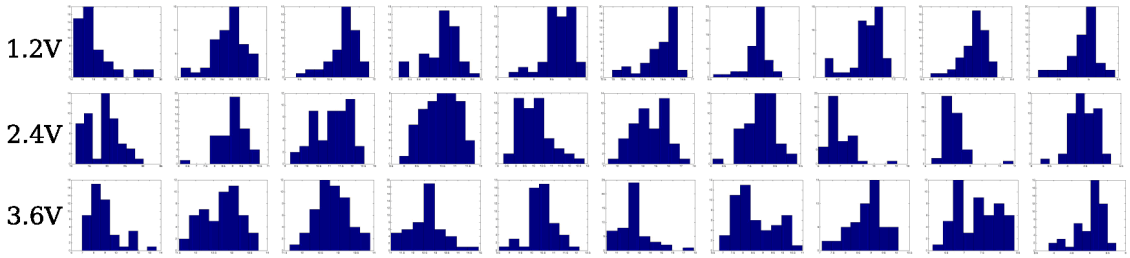

Figura 3.2 – Histogramas de cada banda para 1.2V, 2.4V y 3.6V

La mayoría de los histogramas representan una distribución normal o muy parecida, este hecho nos permite asumir tal distribución y tener así las herramientas estadísticas para el tratamiento de la información.

Una vez obtenidas medias y desviaciones estándar para cada banda, estas se guardan automáticamente en un archivo para poder volver a usarlas sin tener que pasar por el método de aprendizaje. Hay que tener en cuenta que estos resultados dependen tanto del entorno donde se capturaron las señales como del hardware utilizado y, por ello, es posible que no puedan ser utilizados correctamente en otros entornos o máquinas.

3.3.1.1 - Cálculo de la desviación estándar (Sigma)

Para ello se ha utilizado la siguiente fórmula con todos los datos del aprendizaje, en cada banda se aplica:

$$\sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

Figura 3.3 – Calculo de la desviación estándar

En la *Figura 3.3* podemos ver:
$\overline{x}$: media aritmética del grupo de muestras de la banda de frecuencias que se analiza.
n:el numero de muestras de nuestro experimento.
x_i: la muestra "i" del experimento.
s: sigma, o desviación estándar.

Este cálculo se realiza sobre cada grupo de muestras en cada una de las bandas de la señal por separado.

3.3.1.2 - Calculo de la media

El cálculo de la media aritmética se realiza mediante la siguiente formula:

$$\bar{x} = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{a_1 + \cdots + a_n}{n}$$

Figura 3.4 – Media aritmética

Figura 3.4:
$\overline{x}$: media aritmética del grupo de muestras de la banda de frecuencias que se analiza.
a_i: muestra "i" del experimento.

Con la media y la sigma ya se puede calcular si los nuevos espectros son semejantes a los aprendidos o no.
Todos los cálculos se realizan a partir de porcentajes de cada banda de frecuencia, obteniendo diversas muestras según el número de veces que se haya repetido la adquisición de datos a la hora de aprender la señal, cada banda del espectro es dividida

entre el sumatorio de todas las bandas y después multiplicada por cien para conseguir dicho porcentaje.
De esta manera no sólo se mide el peso general de una banda sino también el peso relativo a la señal entera. Y se evitan problemas cuando la señal tiene un poco mas o un poco menos de energía. Como dato negativo podría darse el caso de identificar una señal no valida como una de las clases, pero esto es muy poco probable y menos con el algoritmo que se ha desarrollado, en el siguiente apartado se verán los resultados.

3.3.2 - Método de evaluación

El método de evaluación utilizado es compuesto, por una parte tenemos la evaluación de la señal para cada una de las clases aprendidas y por otra este mismo cálculo repetido para diferentes multiplicadores de la desviación estándar.
En primer lugar hay que explicar el método por el cual se comparan los espectros de las señales.

La primera idea a la hora de comparar los espectros fue comparar cada frecuencia – del espectro - de la señal aprendida con la nueva señal, pero esta solución tenia varios problemas. En primer lugar se tenían que calcular y guardar muchos datos – más de 20000 en el caso de 1 segundo a 22050Hz -.
El volumen de datos no es un problema muy importante pero si había otro problema el desfase de las frecuencias, no siempre están exactamente en las mismas frecuencias, de modo que los datos no acababan de coincidir nunca en una misma frecuencia debido a estas variaciones. Este último inconveniente es solventado de la siguiente forma: utilizando 10 bandas de frecuencia que cubren todo el espectro. De esta forma las variaciones no se notan apenas debido a la concentración en bloques, este método es más robusto pero se pierde en precisión, aunque en entornos controlados no hace falta tanta precisión para discernir las señales.

Para cada una de las bandas en cada clase tenemos su media y su sigma, si la señal a evaluar esta dentro del intervalo [media-sigma*tolerancia, media+sigma*tolerancia] se le da un punto, de esta manera tendremos una puntuación de 0 a 10 para cada una de las clases aprendidas -puesto que hay 10 bandas-.
El parámetro tolerancia es variable tal y como veremos en el algoritmo.

3.3.2.1 - Algoritmo de clasificación de señales

En este subapartado se intentará explicar el algoritmo utilizado para la clasificación de una señal acústica nueva al ser probada en nuestro sistema.

Este algoritmo utiliza un multiplicador de sigma –desviación estándar- variable -a partir de ahora MSigma-, el cual se inicializa a 3, ya que nos asegura un 99% de muestras de la distribución.
Mientras MSigma no sea menor que 0.5 se irán ejecutando los siguientes pasos.
Se compara la señal con cada una de las clases aprendidas mediante un test de hipótesis, esto nos dará 3 resultados numéricos R1, R2 y R3.
Una vez llegado a este punto se calcula cual de R1, R2 y R3 es el mayor y se suma un

punto a la puntuación global G1, G2 o G3 dependiendo del mayor. –Estas puntuaciones comienzan en 0-
Si R1, R2 y R3 no empatan en valores y ninguno es 10 , se termina el algoritmo devolviendo G1, G2 y G3 como puntuaciones finales.
Esto se hace porque normalmente cuando una señal gana con 10 puntos es muy posible que sea por una desviación estándar muy alta de modo que se procede a otra iteración del algoritmo para comprobarlo. Del mismo modo si hay empate no se termina el algoritmo ya que no se puede decidir que a que clase pertenece la señal.
Si Msigma és 3 –primera iteración – y ninguna puntuación supera los 5 puntos estamos ante un claro caso de una señal que no pertenece a nuestras clases aprendidas y por tanto procedemos a la terminación del algoritmo devolviendo G1, G2 y G3 a 0.
Si Msigma es mayor de 0.5 se decrementa en 0.5 y se vuelve a hacer una nueva iteración del algoritmo.

Las variables G1, G2 y G3 son las que se muestran en la casilla "Puntuaciones" de la aplicación.

Pseudocódigo del algoritmo

Se inicializa el msigma a 3.
Bucle mientras msigma > 0.5
Evaluar la señal con cada patrón aprendido.
Si no hay empate entre patrones
    Sumar un punto a la puntuación global del patrón con puntuación en este paso del bucle mas grande.
Fin si

Si no hay empate y ninguna puntuación es 10 se termina el bucle.
Si msigma == 3 y ninguna puntuación supera los 5 puntos se termina el bucle.
Decrementar msgima en 0.5
Fin bucle

En el *Apéndice 7.1* se puede consultar el diagrama de flujo del algoritmo y en el *Apéndice 7.2.3* se puede consultar el código fuente en C++.

De esta manera tenemos una aproximación progresiva -diferentes multiplicadores de sigma –la desviación estándar- para hacer la normal más precisa- y se puede ajustar más la exactitud de los resultados, ya que con mayor probabilidad, dada la distribución normal, los valores se acercaran más al punto medio, es decir, con un multiplicador de sigma menor.

El motivo de esta aproximación es que en las primeras pruebas utilizando únicamente una evaluación de las bandas con multiplicador de la desviación estándar igual a 3, las clases con una alta variación a veces se detectaban erróneamente las clases o incluso se detectaba una señal como perteneciente a dos clases a la vez.
De esta manera se afina más en la detección, acercando más cada vez a la media los valores al decrementar el multiplicador y se realiza la puntuación en varios casos en lugar de uno solo.

3.4 - Manual de uso de la aplicación

En la *Figura 3.5* se puede ver la vista general de la aplicación. Esta dividida en dos partes: la superior, donde se almacenan tanto las clases aprendidas como sus espectros y la inferior, donde se captura la señal a probar en el sistema.

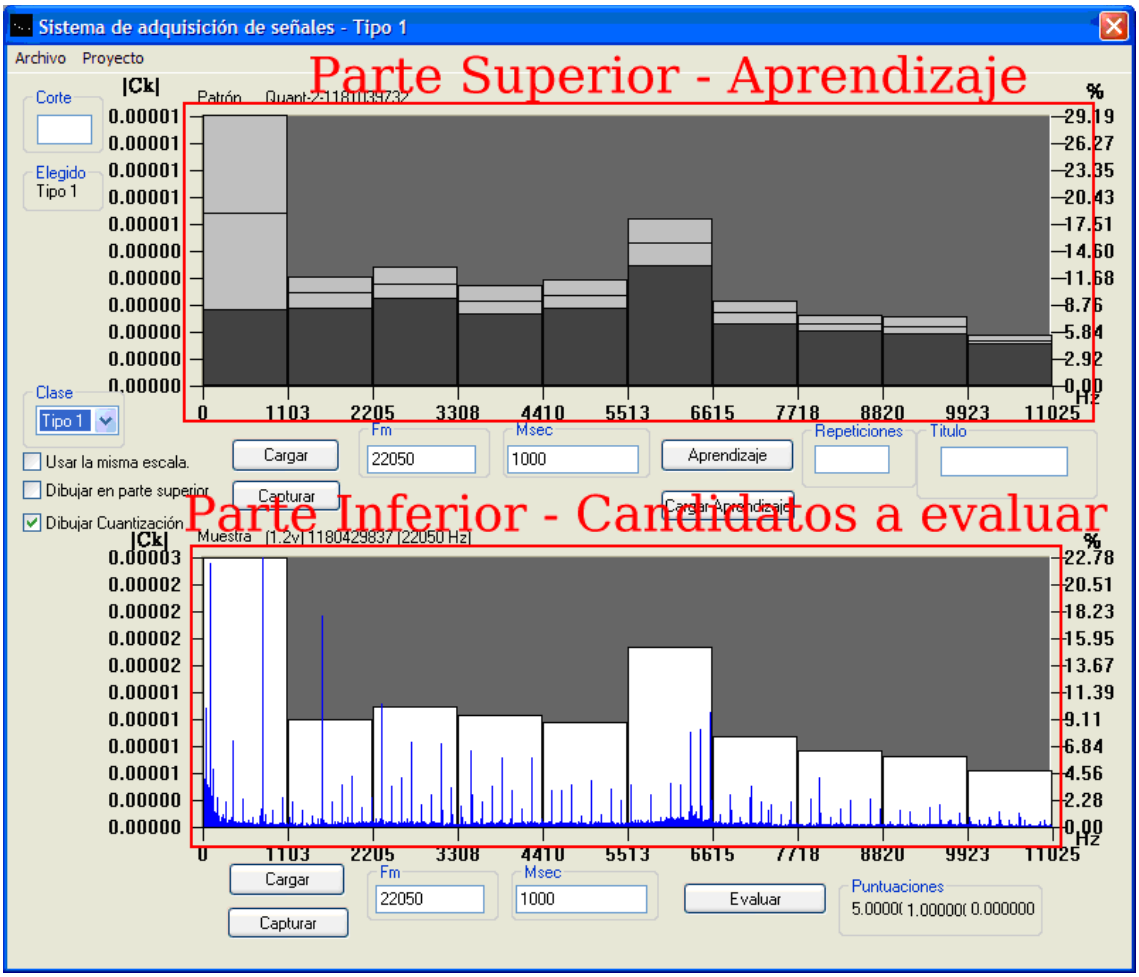

Figura 3.5 – Vista general de la aplicación y de sus partes principales

3.4.1 - Opciones en la parte superior de la aplicación

Figura 3.6 – Opciones de la parte superior de la aplicación

-*Cargar*
Permite cargar de un archivo el espectro de una señal previamente capturada.

-*Capturar*
Captura una señal y calcula su espectro utilizando la frecuencia de muestreo y el tiempo especificados en los recuadros Fm y Msec.

-*Aprendizaje*
Realiza un estudio de la señal capturándola tantas veces como el número indicado en la casilla repeticiones.

-*Cargar Aprendizaje*
Permite cargar un aprendizaje previamente calculado.

-*Cambiar*
Con el botón cambiar y la "Listbox" superior, se puede navegar a través de las 3 posibles clases que se almacenan en la aplicación.

3.4.2 - Opciones en la parte inferior

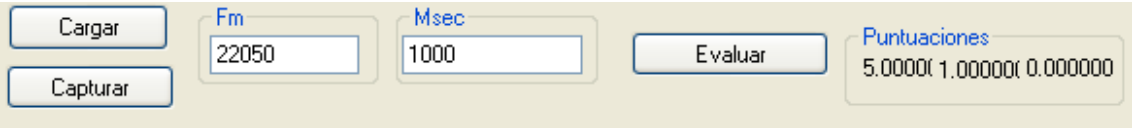

Figura 3.7 – Opciones de la parte inferior de la aplicación

-*Cargar*
Permite cargar una señal previamente capturada.

-*Capturar*
Captura una señal y calcula su espectro utilizando la frecuencia de muestreo y el tiempo especificados en los recuadros Fm y Msec.

-*Evaluar*
Analiza la señal que esta cargada actualmente en la parte inferior con las 3 clases memorizadas en la parte superior y escribe los resultados en las casillas de la parte inferior derecha, de esta manera sabemos a que clase pertenece la señal inferior observando la mayor puntuación, en caso de ser todas cero, significa que la señal no pertenece a ninguna de las aprendidas.

3.4.3 - Opciones globales

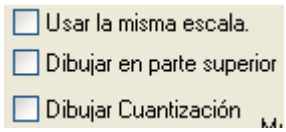

Figura 3.8 – Opciones laterales aplicables a toda la aplicación

-*Usar la misma escala*
Para poder comparar de forma visual las señales con mayor precisión, esta opción permite que ambas graficas se dibujen en la misma escala, utilizando el valor más grande de ambas señales.

-*Dibujar en parte superior*
Esta opción solo se puede activar si ambas señales tienen la misma frecuencia de muestreo. Dibuja ambas señales en la parte superior, para poder compararlas mejor, además se activa automáticamente la opción "Usar la misma escala".

-*Dibujar Cuantización*
Dibuja la Cuantización de los espectros tras estos, ya que a la hora de comparar señales, se utiliza su energía en 10 bandas, no para cada componente Ck.

-*Corte*

Figura 3.9 – Opción para especificar valor máximo en la gráfica

Este recuadro sirve para dibujar los espectros solo hasta la magnitud especificada en él, de este modo se obtiene un mayor detalle de las zonas inferiores de la señal.

-*Titulo*

Figura 3.10 – Titulo de archivos

El texto escrito en el recuadro de título se añadirá al nombre del archivo capturado por la parte delantera, de este modo se pueden crear fácilmente "sets" de pruebas para que sean identificados rápidamente mirando su nombre de archivo.

4. Ejemplo práctico de la aplicación

Una vez definida tanto la teoría como su uso en la aplicación creada, lo único que queda es probarlo con un ejemplo practico, el ejemplo elegido para este proyecto ha sido la identificación del voltaje de un motor de corriente continua – de 0 a 12V – mediante el análisis espectral de la señal que representa el sonido que emite.
Este capítulo expone tanto los elementos necesarios como el método de experimentación y el uso correcto de la aplicación desarrollada para su utilización con un caso real.

4.1 - Experimentación

El ejemplo práctico a realizar será la detección del voltaje de un motor de corriente continua utilizando tres voltajes 1.2, 2.4 y 3.6V, para ello se realiza el aprendizaje de los 3 voltajes en la aplicación y después se entran señales acústicas al sistema para comprobar su evaluación en él.

4.2 - Requisitos Físicos

Más que requisitos propios del proyecto, son requerimientos para la experimentación, ya que el programa final podrá ser utilizado en otros entornos dependiendo de las necesidades de cada situación.

4.2.1 - Elementos utilizados

-Motor DC

El motor de corriente continua es la pieza clave de la experimentación, será utilizado para la realización de las pruebas en tres diferentes voltajes. Se trata de un motor de 0-12V de inducción magnética.

-Pilas 1.2V recargables

Para poder utilizar 3 tipos de voltajes se utilizarán pilas recargables en paralelo, ya que de este modo sus voltajes se suman. Estas pilas son AA estándar de 2000mA.

-Micrófono

El micrófono es el hardware encargado de transformar las vibraciones del aire en voltajes analogicos.
La señal adquirida por el micrófono es de 0 a 1.5 V.

Características generales del micrófono:

• Respuesta de frecuencia: 100 Hz -16 kHz
• Sensibilidad: -67 dBV/µBar, -47 dBV/Pa +/-4 dB
• Voltaje de fuente de alimentación de micrófono: 1,5 V (CC)
• Impedancia: < 1.000 ohmios

-Tarjeta de sonido

La necesidad de capturar la señal acústica requiere de un hardware para su adquisición, como estamos trabajando con PCs la forma más sencilla es mediante la tarjeta de sonido y un micrófono, es importante remarcar que estas dos piezas son claves, ya que dependiendo del micrófono y la tarjeta de sonido, las señales tendrán diferencias.

La tarjeta de sonido se encarga de transformar la señal del micrófono mediante el conversor Analógico/Digital. Los dos parámetros más importantes a la hora de configurar el muestreo de datos son los bits por muestra, que definirán la precisión de los datos cuantizados dando más precisión a mayor número de bits. Y la frecuencia de muestreo que define cuantas muestras por segundo se capturarán de la entrada de audio.

Características de la tarjeta de sonido:

• 8 o 16 bits de precisión
• 1-44100Hz de frecuencia de muestreo seleccionable por software.

4.3 - Procedimiento experimental

El procedimiento experimental nos describirá todos los pasos seguidos para la realización del experimento en si, una vez ya conocidos los elementos que lo componen.

4.3.1 - Experimentación

De cara a la experimentación, se grabó el sonido de un motor en tres velocidades: 1.2V, 2.4V y 3.6V durante un minuto para poder probar el programa a posteriori sin tener que utilizar el mismo, de este modo poniendo la grabación se realizaron los tests del programa.

Antes de empezar la adquisición de las señales acústicas, hay que determinar dos parámetros esenciales, el tiempo de muestreo y la frecuencia de muestreo.
El tiempo de muestreo es arbitrario, pero no sería bueno elegir un tiempo muy bajo ni un tiempo muy alto, si la señal que queremos analizar es continua con un tiempo bajo sería suficiente, en el caso de este experimento se eligió 1 segundo.
El caso de la frecuencia de muestreo es totalmente diferente, hay que elegir una frecuencia de muestreo que evite el *aliasing* de la señal.
El aliasing es un efecto que se produce cuando muestreamos a una frecuencia de muestreo mas baja que el doble de la frecuencia de la señal, tal como dice el teorema de Nyquist.

Podemos ver el aliasing en las *Figuras 4.1 y 4.2*

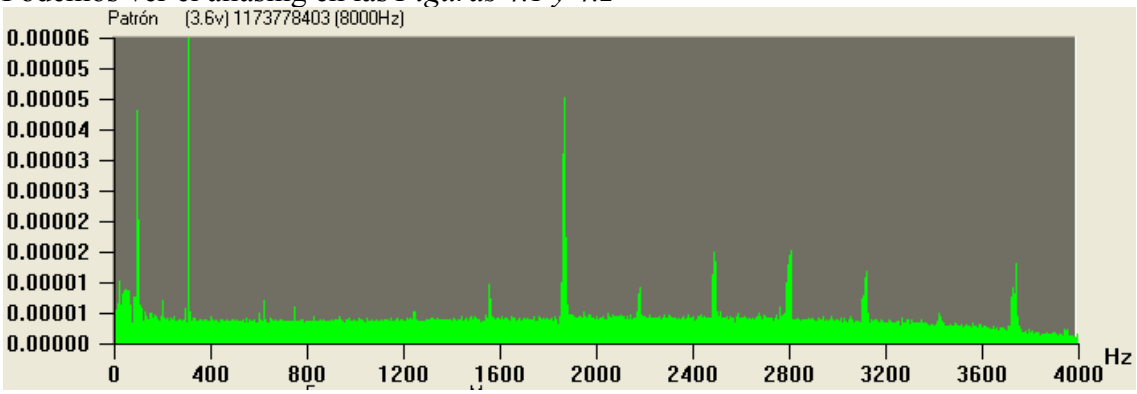

Figura 4.1 – Espectro muestreado a 8000Hz

Figura 4.2 – Espectro muestreado a 44100Hz

En el primer caso las frecuencias se han superpuesto y no se pueden diferenciar, este efecto donde frecuencias altas se representan en frecuencias mas bajas debido a la frecuencia de muestreo es el denominado aliasing. Para evitarlo hay que aumentar la frecuencia de muestreo, tal como vemos en la figura 4.2, en ella se ha eliminado el aliasing pudiendo ver el espectro real de la señal, además observando esta ultima podemos ver que a partir de 11025Hz el espectro contiene valores poco significativos, de modo que podemos elegir una frecuencia de muestreo de 22050Hz –el doble según Nyquist – para nuestro experimento, de esta manera evitamos el aliasing.

En las siguientes figuras podemos ver los espectros de las señales sonoras generadas por el motor a 1.2V, 2.4V y 3.6V y muestreadas a 44100Hz

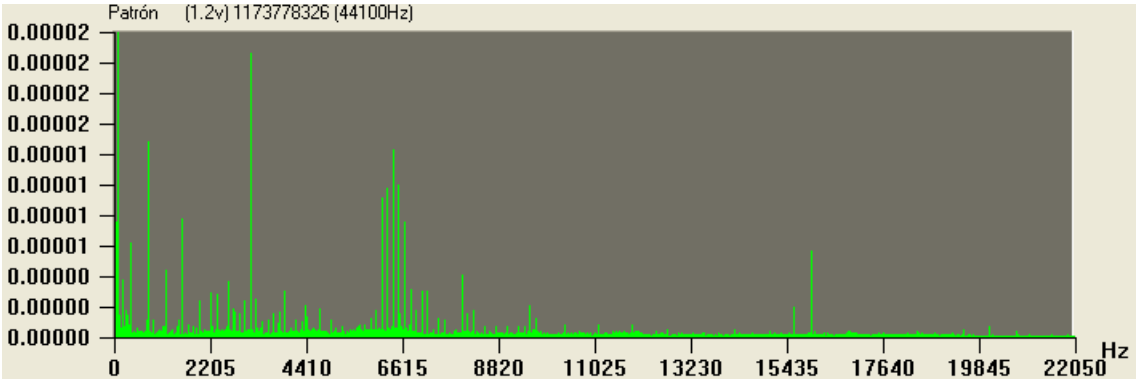

Figura 4.3 – Espectro de la señal acústica del motor a 1.2V (magnitud de |Ck|)

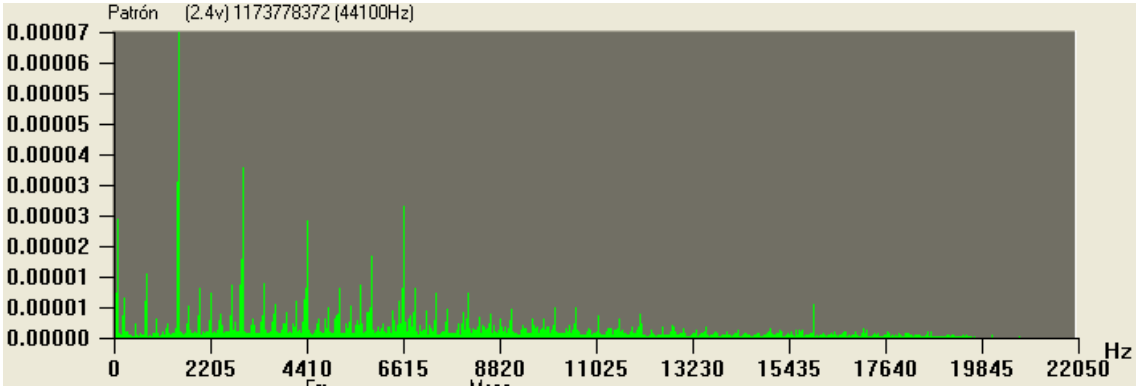

Figura 4.4 – Espectro de la señal acústica del motor a 2.4V (magnitud de |Ck|)

Figura 4.5 – Espectro de la señal acústica del motor a 3.6V (magnitud de |Ck|)

El eje de abscisas solo llega a 22050 porque la frecuencia máxima reconocida es siempre la mitad de la frecuencia de muestreo según el teorema de Nyquist.
A partir de las figuras se contempla que entre las frecuencias de muestreo: 8000, 11025, 22050 y 44100 se pueden desechar 8000 y 11025 ya que hay componentes importantes que no llegaríamos a muestrear al superar estos los 6000Hz, sin embargo se puede observar que no hay frecuencias significativas a partir de 11025Hz por lo que al final nos decantamos por 22050 Hz de frecuencia de muestreo para ahorrar espacio y tiempo de cómputo. Utilizando Nyquist comprobamos que 22050>=2*11025, por lo tanto podemos utilizar dicha frecuencia para nuestros experimentos.

4.3.2 – Metodología

A continuación se describen los pasos necesarios para repetir el experimento con el motor de corriente continua, estos pasos se han realizado cada vez que se quería realizar nuevamente el experimento, para cada una de las pruebas.

-Se conecta el motor con las pilas para conseguir el voltaje deseado.
-Se espera a la estabilización del mismo.
-En un lugar estable, se utiliza el micrófono para grabar el sonido durante 60 segundos.
-Se repite con todos los voltajes.

4.3.3 - Uso de la aplicación

Una vez explicada la manera de realizar el experimento, ya solo queda explicar como utilizar la aplicación para poder realizar el aprendizaje y la clasificación de las señales.

-Se seleccionan en la casilla “Repeticiones” el número de veces a realizar el experimento con una misma clase para su aprendizaje y se pulsa el botón “Aprender”.
-Una vez terminado se elije una clase diferente en la casilla “Clase” y se repite la operación anterior sin necesidad de seleccionar el numero de repeticiones.
-Se prepara el micrófono con la señal a probar contra el sistema y se pulsa el boton “Capturar” de la parte inferior de la aplicación.
-Se pulsa el botón evaluar y se observan los resultados en la casilla “Puntuaciones”.

4.4 - Resultados

Una vez realizado el aprendizaje se pasó a las pruebas, realizando la evaluación de 10 muestras para cada voltaje del motor.

Cada fila de la tabla representa la evaluación de un espectro capturado y cada columna la puntuación de dicho espectro en cada una de las tres posibles clasificaciones de la señal.
La columna con más puntos representa la clasificación elegida, en caso de ser todas cero, significaría que la señal evaluada no pertenece a ninguna de las señales aprendidas.

Los puntos de las columnas han sido elaborados mediante el algoritmo creado en este proyecto, identificando a través de varias etapas con diferentes multiplicadores de sigma, la clase que más se parecía al espectro de la nueva señal acústica adquirida.

4.4.1 - Señales del motor a 1.2V

En la Figura 4.3 y la tabla adjunta se puede observar la distribución de las bandas de energía para el espectro de la señal sonora del motor a 1.2V, se puede ver como la banda dominante es la de bajas frecuencias – [0,1103] Hz -, no solo con alta media sino también con alta variación.

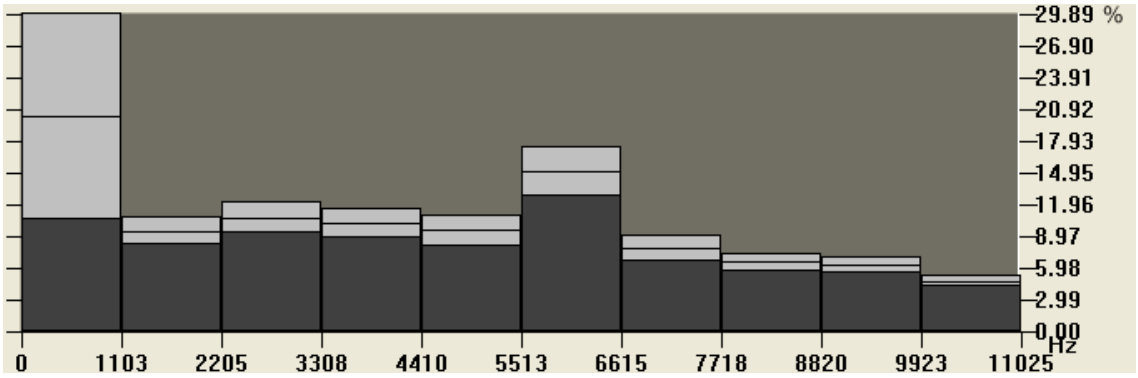

Figura 4.3 – Aprendizaje de la señal de 1.2V

1.2V	Media (%)	Sigma (%)
Banda 1 – [0,1103]Hz	20.1472	3.2478
Banda 2 – [1104, 2205]Hz	9.43357	0.428665
Banda 3 – [2206, 3308]Hz	10.6913	0.496258
Banda 4 – [3309, 4410]Hz	10.103	0.462105
Banda 5 – [4411, 5513]Hz	9.44261	0.482883
Banda 6 – [5514, 6615]Hz	15.0364	0.781569
Banda 7 – [6616, 7718]Hz	7.75227	0.410467
Banda 8 – [7719, 8820]Hz	6.4771	0.280804
Banda 9 – [8821, 9923]Hz	6.19925	0.277292
Banda 10 – [9924, 11025]Hz	4.71736	0.162756

Puntos 1.2v	**Puntos 2.4v**	**Puntos 3.6v**
5	0	0
5	0	0
4	1	0
5	0	0
4	0	0
2	0	0
5	0	0
2	1	0
4	0	0
6	0	0

En el caso de las señales de 1.2V, podemos observar que la mayoría son elegidos inequívocamente excepto en dos casos, el 3º y el 8º, donde la clasificación de 2.4V recibe 1 punto, en el 8º caso es crítico ya que se acerca mucho a la puntuación -2- de 1.2V, esto es debido a las variaciones de la señal que pueden hacer que en ocasiones se acerque mas su distribución a una u otra clase aprendida.

4.4.2 - Señales del motor a 2.4V

En la Figura 4.4 y la tabla adjunta se puede observar la distribución de las bandas de energía para el espectro de la señal sonora del motor a 2.4V, las bandas tienen una distribución bastante similar y la variabilidad de cada una es alta.

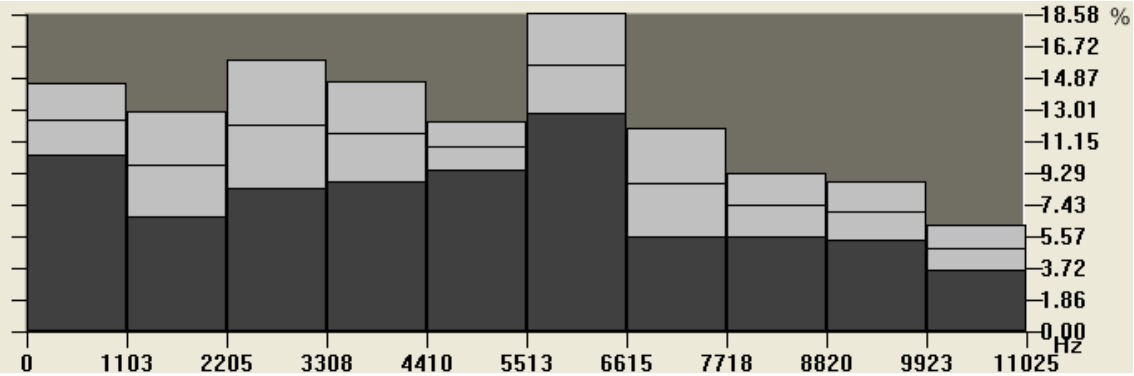

Figura 4.4 - Aprendizaje de la señal de 2.4V

2.4V	Media (%)	Sigma (%)
Banda 1 – [0,1103]Hz	12.365	0.709291
Banda 2 – [1104, 2205]Hz	9.73821	1.03373
Banda 3 – [2206, 3308]Hz	12.082	1.25971
Banda 4 – [3309, 4410]Hz	11.6145	0.991719
Banda 5 – [4411, 5513]Hz	10.8078	0.481755
Banda 6 – [5514, 6615]Hz	15.6133	0.98952
Banda 7 – [6616, 7718]Hz	8.61752	1.07633
Banda 8 – [7719, 8820]Hz	7.3368	0.627017
Banda 9 – [8821, 9923]Hz	6.97424	0.585604
Banda 10 – [9924, 11025]Hz	4.85057	0.457214

Puntos 1.2v	**Puntos 2.4v**	**Puntos 3.6v**
0	0	**1**
0	**2**	1
0	**4**	0
0	**6**	0
0	**4**	0
0	**2**	1
0	**1**	1
0	**3**	0
0	**4**	0
0	**2**	0

Las señales de 2.4V son más problemáticas debido a su similaridad con las de 3.6V, de ahí que consiga puntos la columna perteneciente a los 3.6V e incluso llegue a fallar la identificación de la señal una vez. Esto también es debido a la alta variación de las bandas de 3.6V como podemos ver en la *Figura 4.5*.

4.4.3 - Señales del motor a 3.6V

En la *Figura 4.5* y la tabla adjunta se puede observar la distribución de las bandas de energía para el espectro de la señal sonora del motor a 3.6V, esta señal tiene una variabilidad bastante grande, lo cual puede ocasionar problemas a la hora de detectar las diferentes señales como ya se verá más tarde.

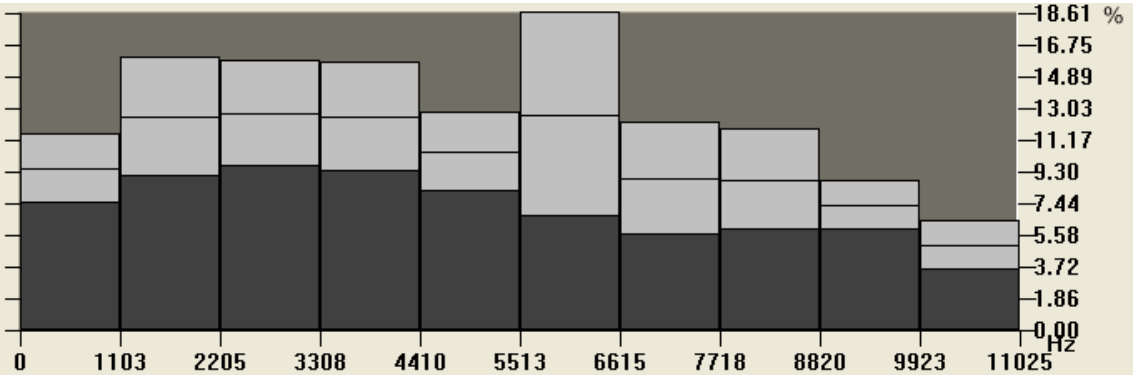

Figura 4.5 - Aprendizaje de la señal de 3.6V

3.6V	Media (%)	Sigma (%)
Banda 1 – [0,1103]Hz	9.43072	0.677356
Banda 2 – [1104, 2205]Hz	12.5024	1.18601
Banda 3 – [2206, 3308]Hz	12.6971	1.05593
Banda 4 – [3309, 4410]Hz	12.472	1.08546
Banda 5 – [4411, 5513]Hz	10.388	0.782908
Banda 6 – [5514, 6615]Hz	12.6166	1.99751
Banda 7 – [6616, 7718]Hz	8.84227	1.11258
Banda 8 – [7719, 8820]Hz	8.81383	1.00735
Banda 9 – [8821, 9923]Hz	7.28923	0.49201
Banda 10 – [9924, 11025]Hz	4.9479	0.49807

Puntos 1.2v	Puntos 2.4v	Puntos 3.6v
0	0	**6**
0	0	**6**
0	0	**6**
0	0	**5**
0	0	**6**
0	0	**4**
0	0	**6**
0	0	**5**
0	0	**6**
0	0	**6**

Las señales de 3.6V tienen un reconocimiento muy sólido, en ningún caso se puntúan otras clasificaciones y la diferencia entre su puntuación y la de las demás clasificaciones es de 4 puntos o superior, lo cual nos indica una gran robustez en su clasificación dado que la puntuación es alta y no hay puntos en otras clases.

4.4.4 - Otras Señales

	Puntos 1.2v	Puntos 2.4v	Puntos 3.6v
cobarde	0	0	0
effect-buzzer	0	0	0
futurebeep	0	0	0
harp	0	0	0
horse	0	0	0
message	0	0	0
truckhorn	0	0	0
whistleshort	0	0	0

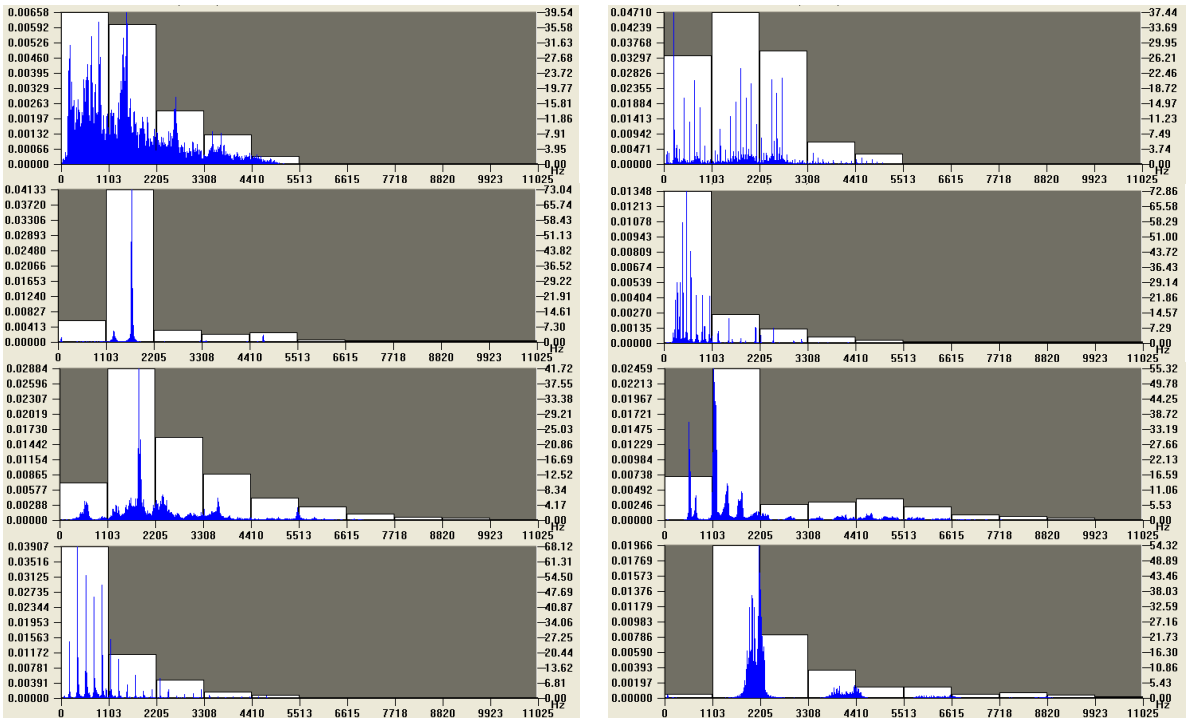

Figura 4.6 – Espectros de las señales de prueba (a la izquierda |Ck| a la derecha %)

La *Figura 4.6* muestra los espectros de las señales elegidas para probar la aplicación, siguiendo el mismo orden de la tabla de izquierda a derecha y de arriba a abajo.

En el caso de las señales de prueba, tenemos ejemplos de sonidos elegidos al azar – desde voz humana hasta la bocina de un camión - . Los espectros en estos casos son tan diferentes que el sistema no tiene ningún problema en su clasificación, de modo que es capaz tanto de detectar correctamente las señales conocidas como de determinar cuales no pertenecen a sus clases aprendidas.

4.5 - Conclusiones del experimento

Los resultados de esta experimentación han sido muy satisfactorios, de 38 experimentos, 37 han sido reconocidos correctamente, lo que nos da un acierto del 97.4%.

Las señales aprendidas han tenido poca interacción entre sí en los resultados, aunque ha habido un fallo en 2.4V siendo clasificado como 3.6V, el echo de tener una desviación estándar (sigma) tan grande en 3.6V ha dado lugar a este suceso. Es posible que al entrenar la clase de 3.6V con más ejemplos esta desviación disminuya, ajustándose más a los datos reales.

Las señales de 1.2V son bastante diferentes al resto y no han tenido problemas para ser identificadas correcta mente al igual que ocurre con las señales ajenas al sistema, estas últimas sirven para verificar que señales externas no sean clasificadas como una de nuestras clases aprendidas.

5. Conclusiones generales

- Se ha estudiado la distribución estadística de los espectros del sonido producido por el motor de corriente continua para elaborar un método para su comparación.

- Se ha diseñado un algoritmo para la detección de señales utilizando múltiples iteraciones y un sistema de puntuación para cada clase aprendida.

- Se ha implementado la FFT en C++ para conseguir el espectro de las señales.

- Se ha aplicado el modelo estadístico de la distribución normal en el reconocimiento de las clases.

- Se ha configurado la tarjeta de sonido para la adquisición de datos mediante el micrófono del PC.

- Se ha diseñado y desarrollado una aplicación para Windows utilizando las MFC que utiliza todos los métodos descritos anteriormente.

- Se ha superado el 95% de reconocimiento de las señales acústicas del motor.

Trabajo Futuro

*Modo contínuo de la aplicación, evaluando todo el tiempo la entrada del micrófono, o en intervalos de tiempo y poder crear calendarios y alarmas.

*Sistema de logs – ficheros con información sobre los resultados del estado de la máquina - para mantener un historial de las evaluaciones.

*Más clases en lugar de 3 únicamente, o que este parámetro fuera seleccionable para cada experimento.

*Posibilidad de guardar los archivos en forma de proyecto, con múltiples experimentos, estadísticas…

6. Referencias Bibliográficas

- *Información sobre la FFT:* **NUMERICAL RECIPES IN C: THE ART OF SCIENTIFIC COMPUTING**. Cambridge University Press.Programs, 1988-1992. ISBN 0-521-43108-5
- *Detección de fallos*: Técnicas para el mantenimiento y diagnostico de maquinas eléctricas. Fernández Cabanas, Manes. MARCOMBO, S.A. 1ª edición. 2000. ISBN 8-426-71166-9
- *Distribución normal:* http://www.fisterra.com/mbe/investiga/distr_normal/distr_normal.htm
- *Varios:* es.wikipedia.org
- *Código fuente para adquisición de datos del micrófono:* http://www.borg.com/~jglatt/tech/lowaud.htm
- *Información FFT:* http://grus.berkeley.edu/~jrg/ngst/fft/fft.html
- *Test de hipótesis:* http://www.terra.es/personal2/jpb00000/ttesthipotesis.htm
- *Test de hipótesis:* http://www.udc.es/dep/mate/estadistica2/sec1_3.html
- *Código fuente FFT:* http://www.yov408.com/html/codespot.php?gg=36

7. Apéndices

7.1 - Diagramas de flujo

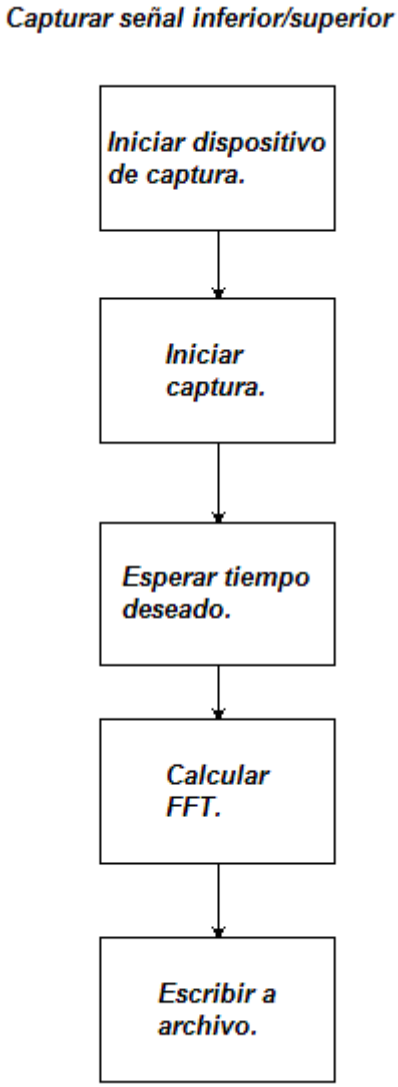

Diagrama 7.1 – Flujo de la captura de señales

El *Diagrama 7.2* explica el algoritmo para capturar una señal acústica desde el micrófono y almacenarlo en la aplicación, sirve tanto para capturar señales en la parte superior de la aplicación como en la parte inferior.

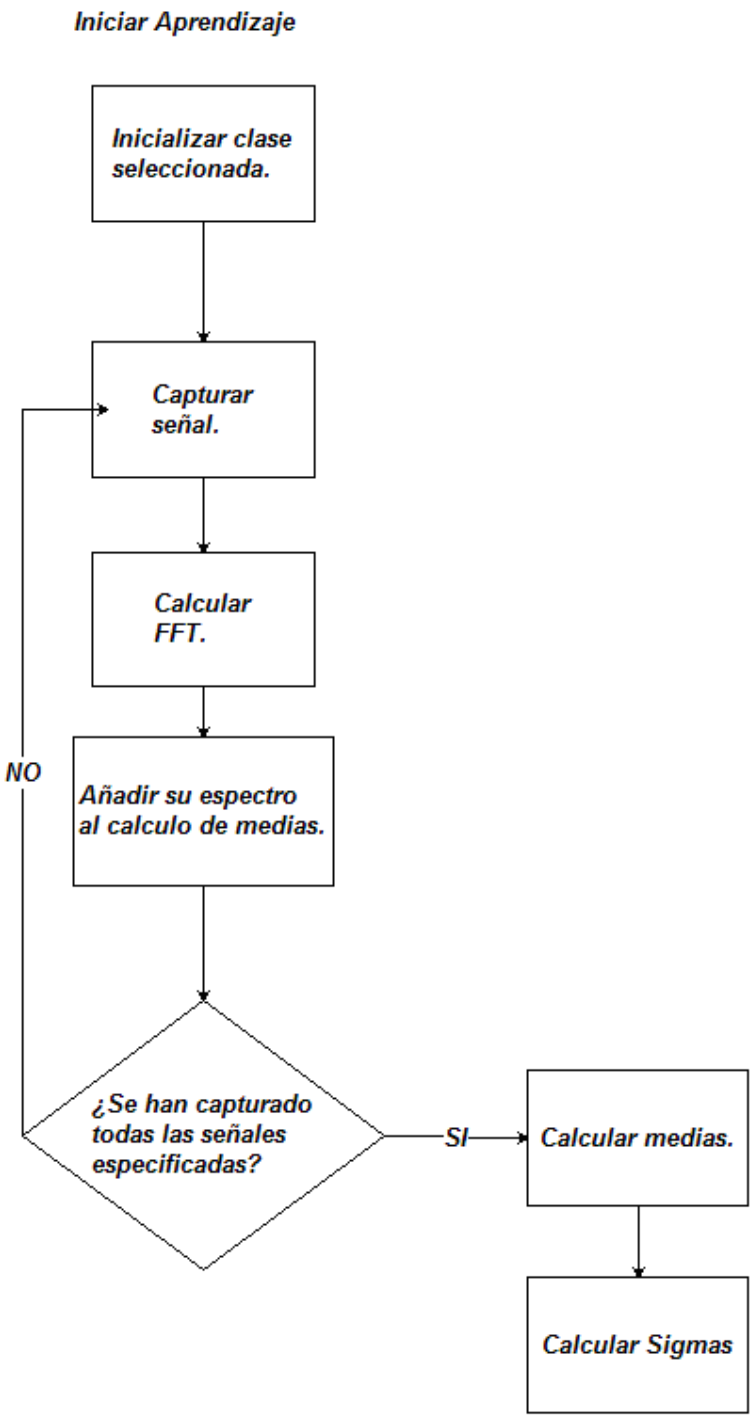

Diagrama 7.2 – Flujo del algoritmo de aprendizaje

El *Diagrama 7.2* muestra el algoritmo de aprendizaje para una señal, se trata de un bucle que almacena todas las señales que necesitamos para posteriormente realizar el cálculo de su media y desviación estándar.

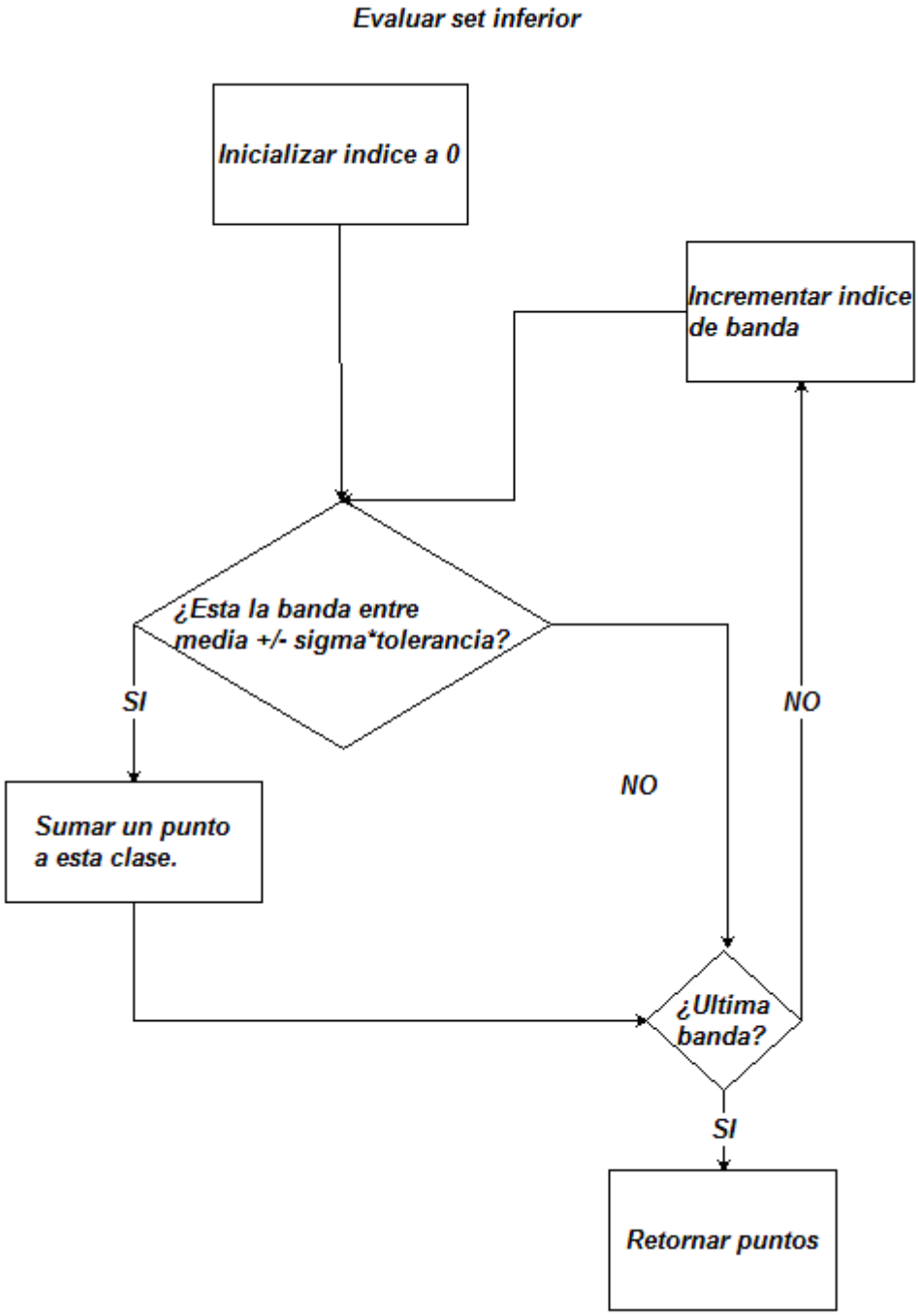

Diagrama 7.3 – Flujo de evaluación de una señal frente a una clase

El *Diagrama 7 .3* representa el algoritmo de evaluación de una señal, se realiza un test de hipótesis sobre cada banda del espectro, retornando el número de bandas que han superado el test de hipótesis.

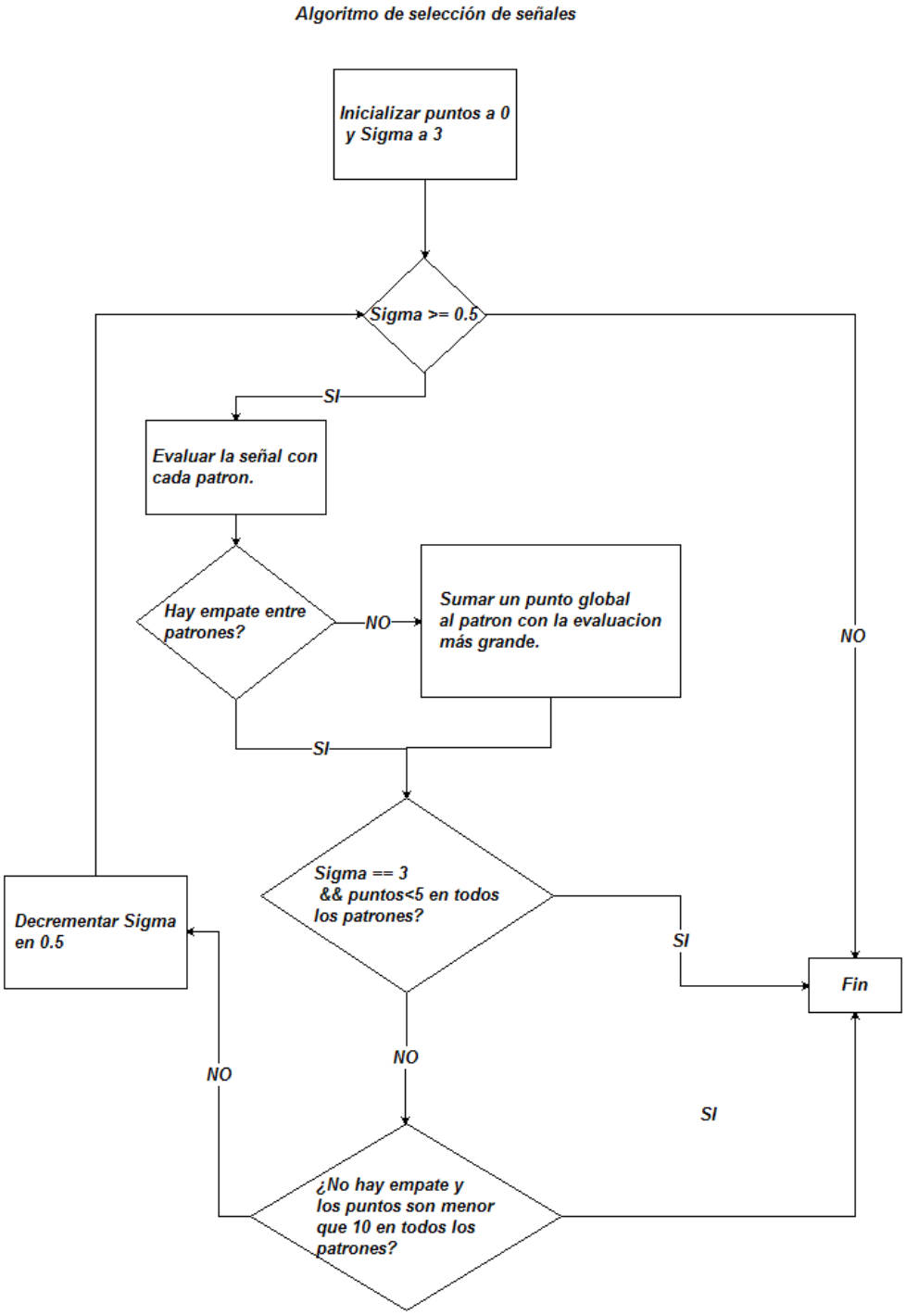

Diagrama 7.4 – Algoritmo de clasificación de señales

Éste algoritmo está explicado en la sección 3.3.2.1

7.2 - Código Fuente

Las porciones de código fuente a continuación están escritas en C++ y forman parte de las funciones más importantes de la realización del proyecto.

7.2.1 - Cálculo del espectro

```
	unsigned NumberOfBitsNeeded ( unsigned PowerOfTwo )
	{
		unsigned i;

		for ( i=0; ; i++ )
		{
			if ( PowerOfTwo & (1 << i) )
				return i;
		}
	}
	unsigned ReverseBits ( unsigned index, unsigned NumBits )
	{
		unsigned i, rev;

		for ( i=rev=0; i < NumBits; i++ )
		{
			rev = (rev << 1) | (index & 1);
			index >>= 1;
		}

		return rev;
	}

void CalculateCk()
	{
	#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

		int isign=1,ir,jr;
		if(Ck)free(Ck);
		Ck=(float *)malloc(sizeof(float)*2*Length);

		unsigned bits=NumberOfBitsNeeded(Length);
		for ( ir=0; ir < Length; ir++ )
		{
			jr = ReverseBits ( ir, bits );
			Ck[(jr<<1)] = Wave[ir];
			Ck[(jr<<1)+1] = 0.0;
		}

		unsigned i, j, k, n;
		unsigned BlockSize, BlockEnd;
```

```
double angle_numerator = 2.0 * 3.141592;
   double tr, ti;
   double delta_angle,sm2,sm1,cm2,cm1,w,ar[3],ai[3];


            BlockEnd = 1;
            for ( BlockSize = 2; BlockSize <= Length; BlockSize <<= 1 )
            {
                  delta_angle = angle_numerator / (double)BlockSize;
                  sm2 = sin ( -2 * delta_angle );
                  sm1 = sin ( -delta_angle );
                  cm2 = cos ( -2 * delta_angle );
                  cm1 = cos ( -delta_angle );
                  w = 2 * cm1;

                  for ( i=0; i < Length; i += BlockSize )
                  {
                        ar[2] = cm2;
                        ar[1] = cm1;

                        ai[2] = sm2;
                        ai[1] = sm1;

                        for ( j=i, n=0; n < BlockEnd; j++, n++ )
                        {
                              ar[0] = w*ar[1] - ar[2];
                              ar[2] = ar[1];
                              ar[1] = ar[0];

                              ai[0] = w*ai[1] - ai[2];
                              ai[2] = ai[1];
                              ai[1] = ai[0];

                              k = j + BlockEnd;
                              tr = ar[0]*Ck[(k<<1)] - ai[0]*Ck[(k<<1)+1];
                              ti = ar[0]*Ck[(k<<1)+1] + ai[0]*Ck[(k<<1)];

                              Ck[(k<<1)] = Ck[(j<<1)] - tr;
                              Ck[(k<<1)+1] = Ck[(j<<1)+1] - ti;

                              Ck[(j<<1)] += tr;
                              Ck[(j<<1)+1] += ti;
                        }
                  }

                  BlockEnd = BlockSize;
            }

            for (i=0;i<Length*2;i++) {Ck[i]=Ck[i]/Length;}
      };
```

7.2.2 - Evaluación de una señal de entrada con una clase en concreto

```
float Evaluate(QSpec *test,float tol,int num)
{
	float *testE;
	int i;
	float puntos=0;
	testE=test->GetEnergy();
	CString hola;
	if(tol==0)tol=3;
	if(Energy && Sigma && testE)
	{
		for(i=0;i<nBlocs;i++)
		{
if(testE[i]>=Energy[i]-Sigma[i]*tol && testE[i]<=Energy[i]+Sigma[i]*tol)
			{
				puntos+=1;
			else
			{
			}
		}

	}
	else return -1;

	return puntos;
}
```

7.2.3 - Evaluar señal de entrada

```
CString stemp;
	float numbers[3],i,puntos[3];
	int j,k,empate;

	for(j=0;j<3;j++)puntos[j]=0;

	for(i=3;i>0;i-=0.5)
	{
		for(j=0;j<3;j++)
			numbers[j]=patrones[j].Evaluate(&qmuestra,i,j);

		/*Mirar si hay empate*/
		empate=0;
		for(j=0;j<3;j++)
			for(k=0;k<3;k++)
				if(j!=k && numbers[j]==numbers[k]) empate=1;

				if(!empate)
				{
					k=0;
					for(j=0;j<3;j++)
						if(numbers[j]>numbers[k]) k=j;

						puntos[k]++;
				}

			for(j=0;j<3;j++)
				for(k=0;k<3;k++)
					if(j!=k && puntos[j]==puntos[k]) empate=1;

	if(!empate && numbers[0]<10 && numbers[1]<10 && numbers[2]<10) break;
	if(i==3 && numbers[0]<5 && numbers[1]<5 && numbers[2]<5) break;

						Sleep(100);
	}
```

RESUM:

En aquest projecte s'ha investigat i desenvolupat una eina per la detecció de fallades mecániques a entorns controlats. Aquest programa analitza l'espectre de les senyals acustiques adquirides mitjançant el microfon del PC i n'estudia estadísticament les propietats –mitjana i desviació estándar- per tal de poder diferenciar les noves senyals que capturi.

S'ha experimentat amb el cas real d'un motor de corrent continu obtenint resultats de mes del 90% d'encert a l'hora de detectar el seu estat.

RESUMEN:

En este proyecto se ha investigado y desarrollado una herramienta para la detección de fallos mecánicos en entornos controlados. Este programa analiza el espectro de las señales acústicas adquiridas mediante el micrófono del PC y estudia estadísticamente las propiedades –media y desviación estándar- para poder diferenciar las nuevas señales que capture. Se ha experimentado con el caso real de un motor de corriente continua obteniendo resultados de más del 90% de acierto a la hora de detectar su estado.

ABSTRACT:

In this project it has been developed and investigated a tool to detect mechanical failures in controlled environments. This program analices the specter of the sound signals adquired through the PC microphone and studies stadistically his properties – mean and Standard deviation- to classify new signals in the future. Experiments with a real DC motor were made obtaining 90% succes detecting it's state.